# Šestnajste vaje iz računalništva I

Simona Peršak

16.3.1999

## 1. Računanje determinante

```
int: matrika, dimenzija, izpuščena  vrstica
out: rezultat
```

Izračunali jo bomo rekurzivno po prvem stolpcu.

### 1.1 Glava funkcije

```
int determinanta(int matrika[MAX_VEL][MAX_VEL],
                 int dimenzija, int izpuscena_vrstica)
```

Funkcija je celoštevilčnega tipa `int determinanta`, ker jo hočemo vrniti kot rezultat.

Potrebujemo podprogram `Prepisi`, ki iz dane matrike izloči 0-ti (v splošnem $j$-ti) stolpec in $i$-to vrstico. Torej ima naslednje parametre:

```
in:   matrika, dimenzija, izloceni stolpec, izlocena vrstica
out:  nova_matrika
```

Glava te funkcije je tako:

```
void prepisi(int matrika[MAX_VEL][MAX_VEL], int dimenzija,
             int izp_stolpec, int izp_vrstica,
             int nova_matrika[MAX_VEL][MAX_VEL])
```

#### 1.1.1 Premakanje stolpcev in vrstic v matriki

V novi matriki imamo 1 vrstico in 1 stolpec manj. Dokler je števec, s katerim se sprehajamo po vrsticah v stolpcu, manjši od izpuščenega vrstice, vrstice prepisujemo, ko pa postane večji, jih pomaknemo za eno navzgor.

Imamo dva kompleta indeksov:

1. za izvorno matriko `matrika`, iz katere prepisujemo:

   ```
   int i, j;
   ```

   in

2. za ciljno matriko `nova_matrika`, v katero prepisujemo.

   ```
   int i_ciljni, j_ciljni;
   ```

S prvim kompletom se sprehajamo po izvorni matriki in z drugim po ciljni. Prvi komplet se vedno povečuje za po ena, drugi pa le takrat, ko elemente prepisujemo (nismo niti na izpuščeni vrstici niti v izpuščenem stolpcu).

## 1.2 Zapis izraza $(-1)^i$ v prog. jeziku c

V jeziku c ne moremo preprosto zapisati tega izraza, zato pogledamo, ali je eksponent sod ali lih in odvisno od tega izraz prištejemo ali odštejemo.

```
if ((i + 0) % 2 = 0) /* sodo */
  rezultat += matrika[i][0] * det(dim - 1, zac_matrika) \\
else                 /* liho */
  rezultat -= matrika[i][0] * det(dim - 1, zac_matrika)
```

## 1.3 Koda

```
void prepisi(int matrika[max_vel][max_vel], int dimenzija,
             int iz_stolpec, int iz_vrstica,
             int nova_matrika[max_vel][max_vel])
{ int i;  /* vrstica */
  int j;  /* stolpec */
  int i_c;  /* ciljna vrstica */
  int j_c;  /* ciljni stolpec */

  for(i = 0, i_c = 0; i < dimenzija; i++) {
    if (i != iz_vrstica) {
      for(j = 0, j_c = 0; j < dimenzija; j++)
        if (j != iz_stolpec) {
          nova_matrika[i_c][j_c] = matrika[i][j];
          j_c++;
        } /* if */
      i_c++;
    } /* if */
  } /* for */
} /* prepisi */

/* ------------------------------------------------------------------ */

int determinanta(int dimenzija, int matrika[max_vel][max_vel])
{
  int rezultat;
  int i;                              /* vrstica */
  int zac_matrika[max_vel][max_vel];  /* zacasna matrika */

  if (dimenzija == 1)  return  matrika[0][0];
  else {
    rezultat = 0;
    for (i = 0; i < dimenzija; i++) {
      prepisi(matrika, dimenzija, 0, i, zac_matrika);
      if ((i + 0) % 2 == 0)
```

```
        rezultat+=
          matrika[i][0] * determinanta(dimenzija - 1, zac_matrika);
      else
        rezultat-=
          matrika[i][0] * determinanta(dimenzija - 1, zac_matrika);
    } /* for */
    return rezultat;
  } /* else */
} /* determinanta */
```

Celoten program je objavljen v novičkah.

# 2. Pri funkcijah ločimo

1. definicijo (opis) funkcije
2. klic funkcije

## 2.1 Definicija funkcije

Definicija funkcije oz. opis je kot recept pri kuhanju. Nima nobenega dejanja je le opis postopka in seznam sestavin, ki sestavljajo določeno jed.

## 2.2 Primer klica funkcije

Funkcija za izračun determinante je opisana v funkciji `determinanta`. Takrat, ko je *poklicana*, prične *obstajati njen primerek*. Podobno, ko po receptu spečemo pico, prične pica obstajati. Seveda ni razloga, da ne bi po istem receptu spekli še ene pice, ki s prvo nima nič skupnega.

Tako tudi zadržka, da ne bi funkcija *poklicala* same sebe in ustvarila še en primerek, ki s prejšnjim nima nič skupnega. Še posebej spremenljivke (in rezultat) v novem stvarjenju funkcije nimajo nič s spremenljivkami in rezultatom v prejšnjem ustvarjenju (predhodne determinante z 0-to vrstico in $j$-tim stolpcem). Med seboj nimata nobene povezave, vsakič je nov rezultat in novi parametri. Rezultati so meds ebojno neodvisni.

Rezultat v `if` stavku je bistven in je v lasti klicane funkcije oz. živeče funkcije.

Pri tem primeru je bila podmatrika seveda definirana kot lokalna spremenljivka.

Primer matrike prikazan s pripravo palačink:

- matrika $\equiv$ recept
- poraba prostora v programu $\equiv$ moka
- nova matrika $\equiv$ za vsako palačinko novo testo

Nove matrike moramo delati, ker se matrike kličejo in s praznimi vrsticami n e znamo delati.

Naučiti se moramo matriko obravnavati kot predmet pri kateri bi se izpuš čala vrstica in stolpec oz. napraviti recept za več palačink hkrati.

Matriko hočemo obravnavati kot celoto. Pojem matrike dopolnimo z dvema vektorjema. Enega imamo v vodoravni smeri, drugega v navpični smeri. V teh vektorjih imamo zastavice, ki so bodisi dvignjene ali spuščene. Dvignjena zastavica pomeni, da je dotična vrstica (oz. stolpec) izpuščen.